DMIF, Università degli Studi di Udine

Invarianti e codifica di Huffman

Emanuele Scapin, Ph.D. Student
scapin.emanuele@spes.uniud.it

16 Giugno 2021

Indice

Una definizione

In matematica un oggetto (funzione, insieme, punto, ...) si dice **invariante** rispetto o sotto una trasformazione se esso rimane inalterato dopo l'azione di tale trasformazione.

Potenza

Dati due valori interi $B > 0$ ed $E \geq 0$, calcolare il valore B^E.

Esempio

```
int z=1;
int y=0;

while (y != e) {
    y = y + 1;
    z = z * b;
}
// z valore della potenza calcolata
```

Cosa cambia e cosa non cambia (invariante)?

Potenza

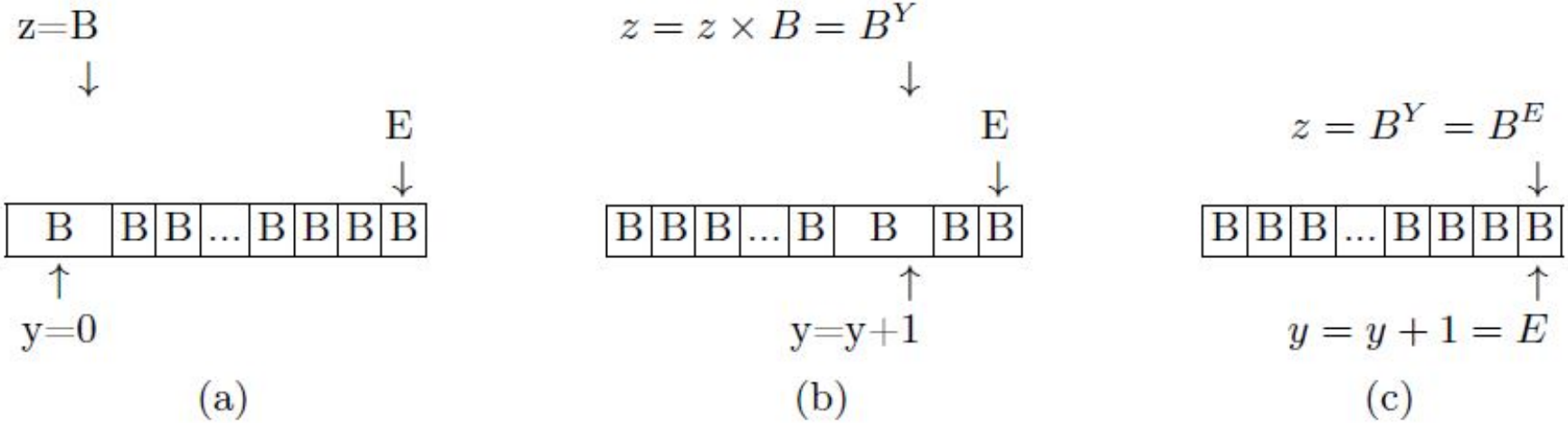
```
int z=1;
int y=0;

while (y != e) {
    y = y + 1;
    z = z * b;
}
```

Due informazioni importanti:

- $Z = B^Y$
- $Y \leq E$

Alcune informazioni importanti:

- $Z = B^Y$
- $Y \leq E$



Somma dei primi N naturali

Dato $N > 0$ calcolare il valore della sommatoria $\sum_{i=1}^{N} i$ senza usare la moltiplicazione.

Esempio

```
int z=1;
int y=1;

while (y != n) {
    y = y + 1;
    z = z + y;
}
// z valore della somma
```

Cosa cambia e cosa non cambia (invariante)?

Somma dei primi N naturali

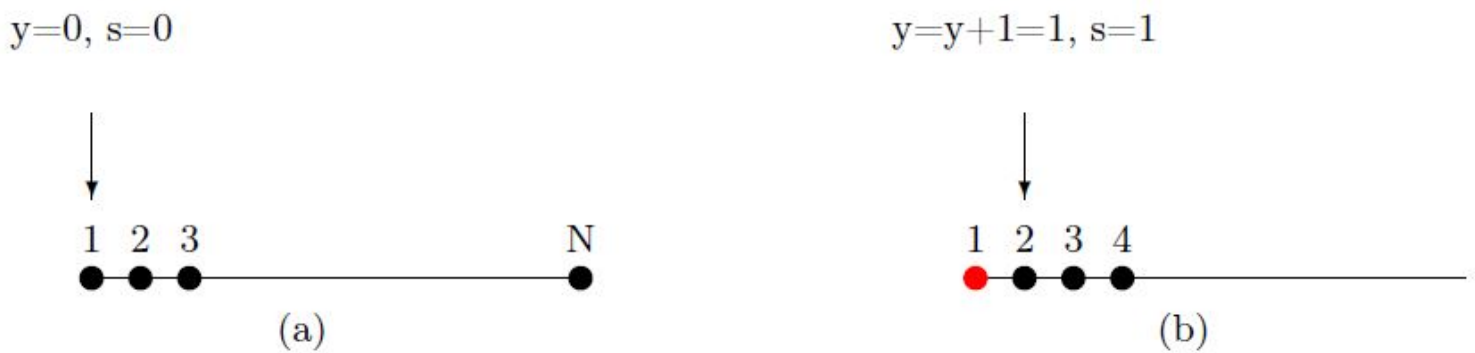
```
int z=1;
int y=1;

while (y != n) {
    y = y + 1;
    z = z + y;
}
```

Alcune informazioni importanti:

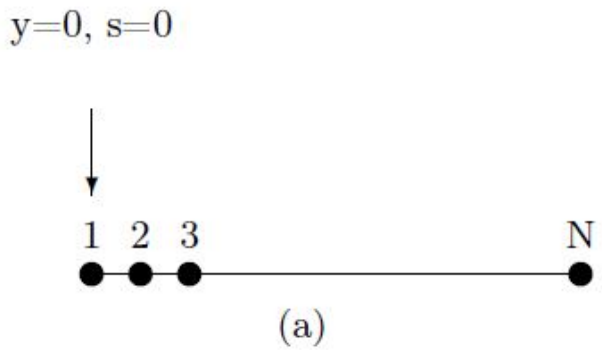
- $Z = \sum_{i=1}^{Y} i$
- $Y \leq N$
- $Y > 0$

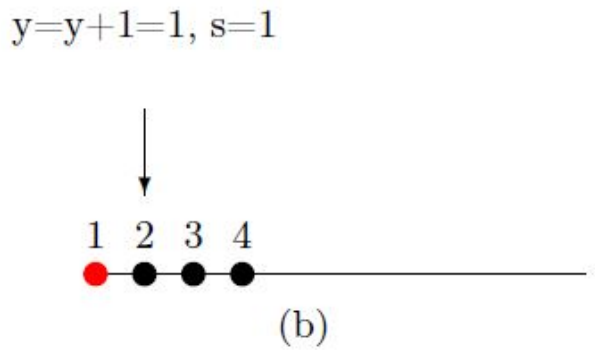
Alcune informazioni importanti:

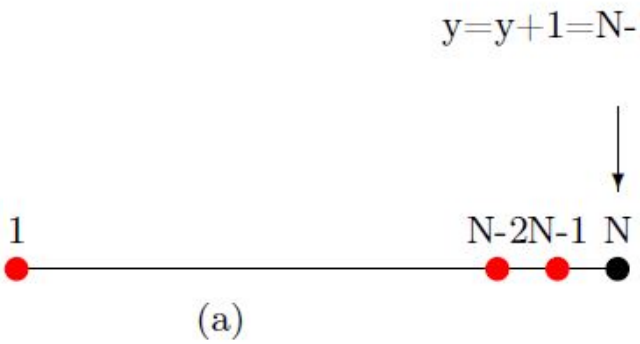
- $Z = \sum_{i=1}^{Y} i$
- $Y \leq N$
- $Y > 0$

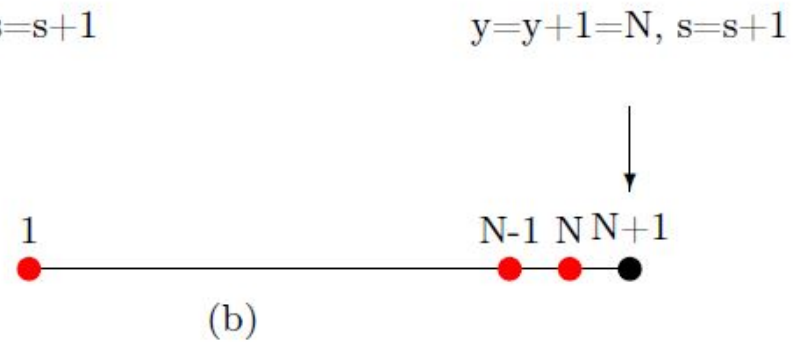


Una rappresentazione efficace.



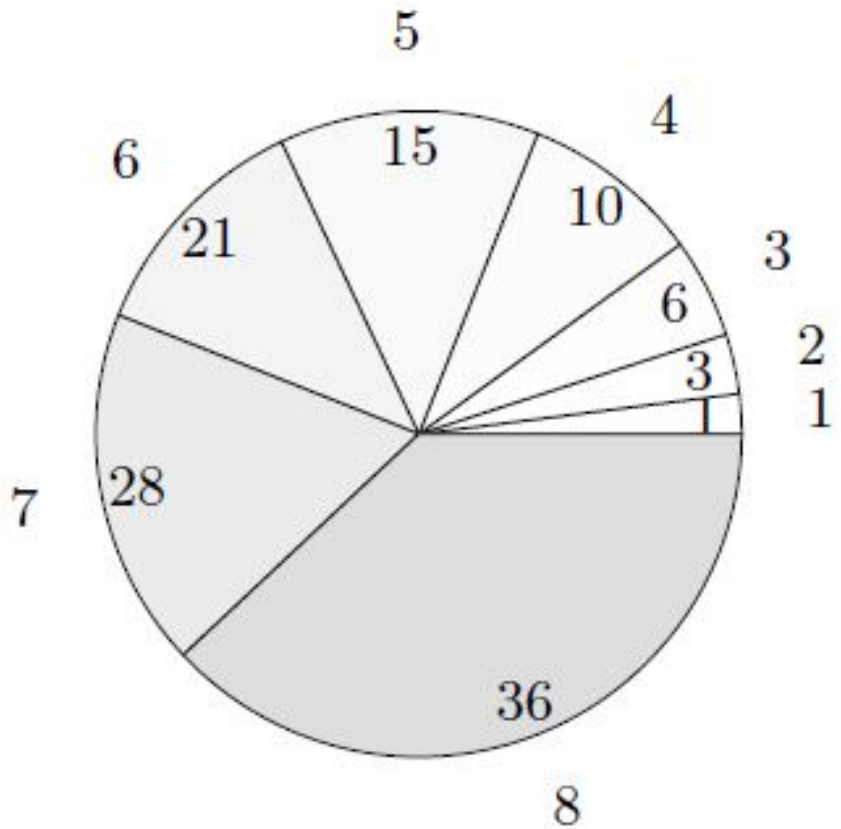






Somma dei primi N naturali
Con immagini

Una rappresentazione poco efficace.



Algoritmo del contadino russo

Prodotto tra due numeri interi $N \geq 0$ e $M \geq 0$, ovvero $N \times M$.

```
int x = m;
int y = n;
int z = 0;

while (y > 0) {
     if (y % 2 > 0) {
          z = z + x;
     }
     x = x * 2;
     y = y / 2;
}

return z;
```

Algoritmo del contadino russo

- $nm = ?$

```
int x = m;
int y = n;
int z = 0;

while (y > 0) {
    if (y % 2 > 0) {
        z = z + x;
    }
    x = x * 2;
    y = y / 2;
}

return z;
```

Algoritmo del contadino russo

- $nm = z+?$

```
int x = m;
int y = n;
int z = 0;

while (y > 0) {
    if (y % 2 > 0) {
        z = z + x;
    }
    x = x * 2;
    y = y / 2;
}

return z;
```

Algoritmo del contadino russo

- $nm = z + xy$

```
int x = m;
int y = n;
int z = 0;

while (y > 0) {
    if (y % 2 > 0) {
        z = z + x;
    }
    x = x * 2;
    y = y / 2;
}

return z;
```

Algoritmo del contadino russo

- passo iniziale: $nm = z + xy = 0 + xy = 0 + nm$, infatti $z = 0$ e $x = m$ e $y = n$.
- passo successivo (se y pari): $nm = z + (2x)(y/2) = z + xy$
- passo successivo (se y dispari): $nm = (z + x) + (2x)(((y - 1)/2) = z + x + x(y - 1) = z + xy$
- passo finale ($y = 0$): $nm = z + xy = z$

```
while (y > 0) {
    if (y % 2 > 0) {
        z = z + x;
    }
    x = x * 2;
    y = y / 2;
}
```

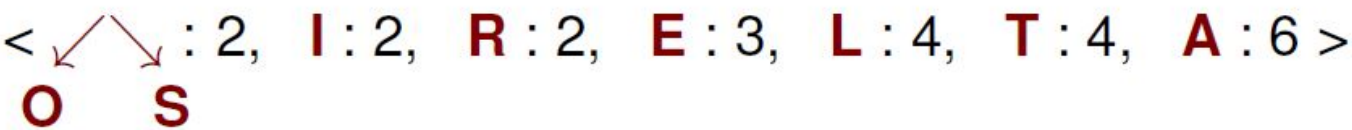
Albero di Huffman
Primo invariante

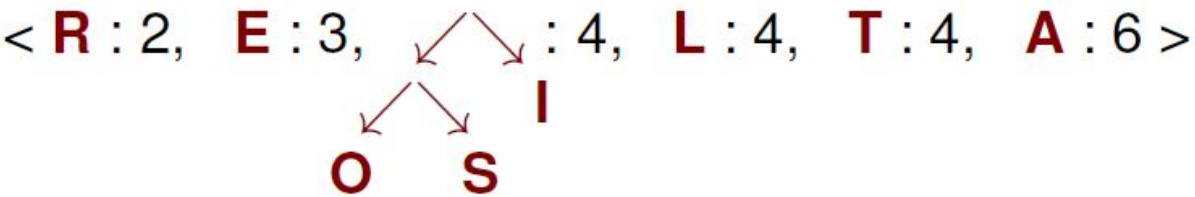
- Per ogni nodo, la somma dei pesi del sottoalbero di sinistra più la somma dei pesi del sottoalbero di destro è il peso del nodo stesso.

  - Lettere e pesi = numeri di occorrenze:

    < **O** : 1, **S** : 1, **I** : 2, **R** : 2, **E** : 3, **L** : 4, **T** : 4, **A** : 6 >

  - Costruzione dell'albero di Huffman:

- Per ogni nodo, la somma dei pesi del sottoalbero di sinistra più la somma dei pesi del sottoalbero di destro è il peso del nodo stesso.

```
public Node( Node left, Node right ) {   // nodi interni

    itm = NO_BYTE;
    wgt = left.weight() + right.weight();
    cst = left.cost() + right.cost() + wgt;
    lft = left;
    rgt = right;

}
```

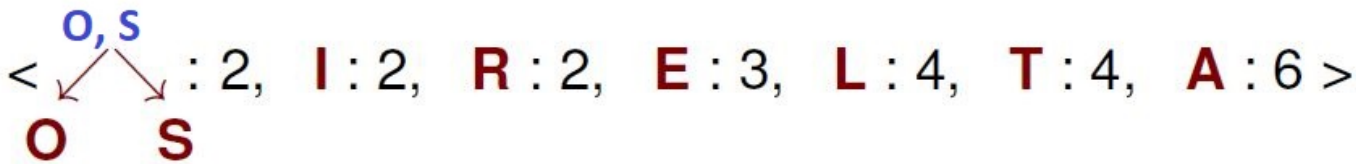
Albero di Huffman
Secondo invariante

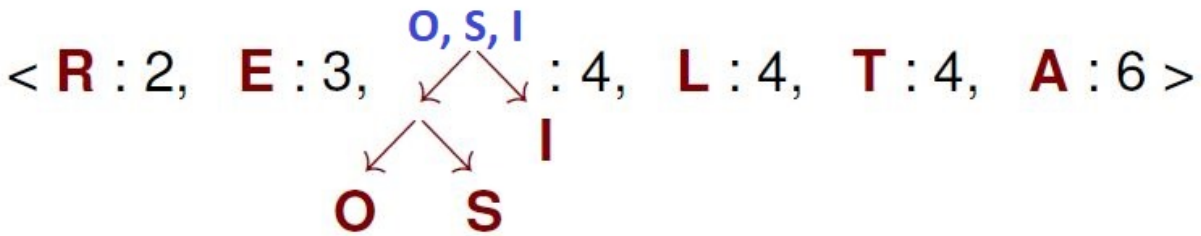
- Per ogni nodo, l'elenco dei simboli del nodo sono tutti i simboli che si trovano nelle foglie del sottoalbero sinistro e tutti i simboli che si trovano nelle foglie del sottoalbero destro.

  - Lettere e pesi = numeri di occorrenze:

    < **O** : 1, **S** : 1, **I** : 2, **R** : 2, **E** : 3, **L** : 4, **T** : 4, **A** : 6 >

  - Costruzione dell'albero di Huffman:

- Per ogni nodo, la somma dei pesi del sottoalbero di sinistra più la somma dei pesi del sottoalbero di destro è il peso del nodo stesso.

Necessarie modifiche a Node

```
public Node( Node left, Node right ) {   // nodi interni

    itm = NO_BYTE;
    wgt = left.weight() + right.weight();
    cst  = left.cost() + right.cost() + wgt;
    lft  = left;
    rgt  = right;
    code = left.code + ", " + right.code; // genero una sequenza
        di caratteri
}
```

Il costo

```
public Node( Node left, Node right ) {   // nodi interni

    itm = NO_BYTE;
    wgt = left.weight() + right.weight();
    cst = left.cost() + right.cost() + wgt; // costo
    lft = left;
    rgt = right;
}
```

- Il **costo** di un albero di Huffman è equivalente alla dimensione del documento quando è codificato utilizzando il codice definito dall'albero.
  In altre parole, se s_i appare o_i volte e s_i è una foglia di profondità d_i nell'albero, allora ogni occorrenza di s_i nel documento è codificata con d_i bit.
  Di conseguenza, s_i contribuisce per $o_i \times d_i$ al costo totale del documento codificato.
  In generale il costo totale del documento codificato (e quindi dell'albero) è dato da
  $o_1 \times d_1 + o_2 \times d_2 + \cdots + o_i \times d_i$

- Questo concetto di costo è applicabile anche ai sottoalberi, in questo caso il costo della radice del sottoalbero considerato ci dice la dimensione del documento che codifica solamente i simboli presenti nel sottoalbero (quindi un documento dove sono stati eliminati i simboli non considerati)
- Questo concetto di costo è applicabile all'albero nella sua interezza (per il documento completo), ma anche ad ogni sottoalbero (per una frazione del documento), ed è invariante.